Project 4: Particle Filter
RBE 595: Advanced Robot Navigation

Miheer Diwan
Robotics Engineering
Worcester Polytechnic Institute
msdiwan@wpi.edu

I. Introduction

In this project, we will implement a Particle Filter to estimate the pose of a quadcopter drone. Pose estimation is a critical task in robotics, enabling precise control and navigation of aerial vehicles. We will leverage sensor data from AprilTags and motion capture systems to improve the accuracy of pose estimation.

II. Pose Estimation

To estimate the pose of the quadcopter drone, we will employ the Perspective-n-Point (PnP) problem-solving approach. This method utilizes the AprilTag corners from the map layout as 3D points in the world and their corresponding 2D projections in the image plane captured by the camera.

The camera calibration matrix and distortion coefficients are given as follows:

$$\text{Camera Matrix} = \begin{bmatrix} 314.1779 & 0 & 199.4848 \\ 0 & 314.2218 & 113.7838 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{Distortion Coefficients} = \begin{bmatrix} -0.438607 & 0.248625 & 0.00072 & -0.000476 & -0.0911 \end{bmatrix}$$

The camera model gives the realtion between the homogenous world coordinates and the homogenous image coordinates and is represented as:

$$\mathbf{M} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

After obtaining the orientation data in the form of a rotation vector, we will convert it to a rotation matrix. Using this rotation matrix, we can then extract the three Euler angles using the provided matrix transformation formula:

$$\begin{aligned} \phi &= \text{atan2}(R_{32}, R_{33}) \\ \theta &= \text{atan2}(-R_{31}, \sqrt{R_{32}^2 + R_{33}^2}) \\ \psi &= \text{atan2}(R_{21}, R_{11}) \end{aligned}$$

With this, we can obtain the transformation matrix from the camera frame to the world frame. Additionally, the provided information for the yaw rotated with $\pi/4$ and the roll of π is used for get the transformation from the camera frame to the IMU frame.

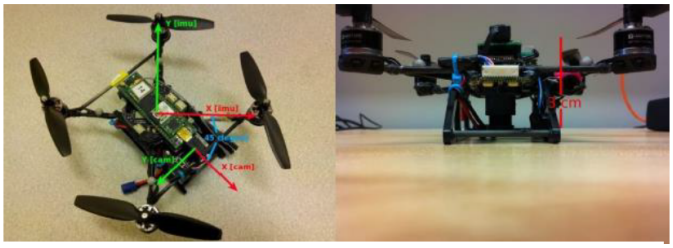


Fig. 1. Drone Configuration

III. Task 1: Particle Filter Implementation

The first step for the particle filter is to generate the particle.The `ParticleFilter` class encapsulates the particle filter algorithm, featuring methods for prediction, update, resampling, and state estimation using a collection of particles.

A. Filter Initialization

Upon initialization, the initial particles are spawned with the following characteristics:

$$\begin{aligned} p &= (0, 3) \quad \text{for position (m)} \\ q &= (-\frac{\pi}{2}, \frac{\pi}{2}) \quad \text{for angles (rad)} \\ b\_g &= (-0.5, 0.5) \quad \text{for gyroscope bias} \\ b\_a &= (-0.5, 0.5) \quad \text{for accelerometer bias} \end{aligned}$$

B. Prediction Step

In the second step, we predict for each state and all the particles created in that state. We vectorize instead of using for loopps to save compute. Additionally, I added noise to the control inputs. The G_q_{inv} and R_q matrices are calculated. This noisy input is used to predict the next state by passing the particle through the process model. The process model adds a noise, sampled from a zero mean gaussian distribution with a covariance Q, to the state prediction.

$$x_{\text{next}} = x_{\text{prev}} + (\dot{x} + \text{noise}) \times dt$$

The noise covariance matrix Q is defined as:

$$Q = \text{diagonal}([0.01]) \times 500$$

C. Weight Update Step

To compute the weights, and the update step, we calculate the error between the measurement z and the predicted particles. The projection of this error in a Gaussian funtions is used to calculate the weights centered around the given measurement z with covariance R. The weights are then adjusted by the highest index and compute a weighted average of the particles and the weights. The covariance matrix R is used by setting the non-diagonal elements to zero.

$$\mathbf{R} = \text{diagonal}([0.00669, 0.00484, 0.00881, 0.00412, 0.00638, 0.00110])$$

D. Low Variance Resampling

The `low_variance_sampling()` method implements the low variance resampling algorithm. Following resampling, particle weights are reset to ensure equal weighting for all particles.

IV. Task 2

A. Navigation solution and particle count investigation

In this task, we evaluate the Root Mean Square Error (RMSE) across three state sampling methods: **weighted average**, **highest weight**, and **simple average**. The particle count was kept consistently at 1000. The primary aim was to scrutinize the fluctuation of RMSE over time, with particular attention to the integration of low variance resampling techniques.

Observations: Since the loss decreases, partile filter algorihtm is robust for state estimation.

- Across the datasets, all three sampling methods exhibited a consistent downtrend in RMSE as data processing unfolded. This diminishing error signifies a tangible enhancement in the precision of state estimates provided by the particle filter in comparison to the ground truth states.
- The incorporation of low variance resampling yielded palpable benefits, and reduces the RMSE. This resampling strategy plays a pivotal role in preserving particle diversity.

B. RMSE for Different Datasets and Particle Sizes

This analysis gives us an insight on how the choice of particle size impacts the precision of the particle filter. We use the 3 sampling methods mentioned above.
The 'final_rmse_result.csv' file with the results is stored in the 'Outputs' folder.

- **Impact of Particle Size:** Notably, increasing the particle size generally results in a reduction of RMSE values, indicative of enhanced accuracy in state estimation. This enhancement can be attributed to the heightened likelihood of capturing the true state with a larger ensemble of particles.
- **Comparison of Sampling Methods:** Among the trio of sampling methods scrutinized—weighted average, highest, and average—subtle differentials emerge. Specifically, the weighted average and average methodologies exhibit a slight edge over their highest counterpart. This variance may emanate from the distinct characteristics inherent in each method; while averaging strategies tend to mitigate estimation disparities, the highest weight particle selection may occasionally rely on outlier estimates, potentially diverging from the authentic state.

V. Task 3

A. Ease of Implementation

1) **Code Writing:** Implementing the Extended Kalman Filter (EKF) was simpler compared to the Particle Filter (PF). EKF relies on linearization techniques, making it conceptually easier to implement. Additionally, abundant resources and libraries for linear Kalman Filters facilitated coding. PF involves complex probabilistic sampling and resampling, requiring deeper understanding and careful implementation.
2) **Parameter Tuning:** Tuning parameters for EKF was simpler compared to PF. EKF parameters like process and measurement noise covariance matrices can often be adjusted based on prior knowledge or trial and error. PF involves tuning parameters like the number of particles and noise covariance matrices, requiring a nuanced understanding and sensitivity to different settings.

B. Speed of Code

1) **Runtime Performance:** EKF typically runs faster than PF. EKF involves simpler mathematical operations, while PF requires particle sampling and resampling steps, leading to higher computational overhead for PF.
2) **Importance:** Speed of code execution is crucial for real-time applications like autonomous vehicles or robotics. Faster execution allows for more responsive systems, enabling timely adjustments to changing conditions.

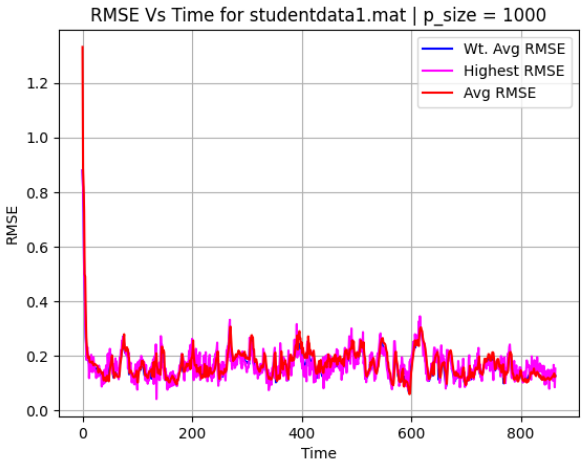
C. Accuracy of Results

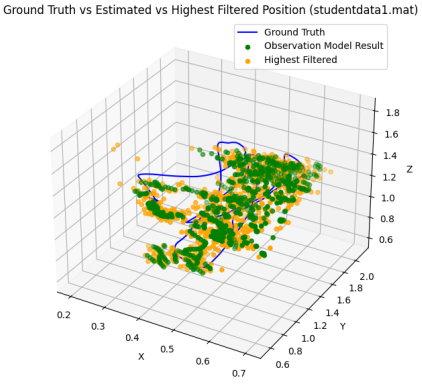
1) **Tracking Accuracy:** PF generally yields more accurate results than EKF, especially in highly non-linear and non-Gaussian systems. PF can represent complex distributions more effectively through particle-based representation, handling non-linearities and uncertainties better.
2) **Factors Influencing Accuracy:** Accuracy depends on factors like system dynamics complexity, sensor measurement quality, and parameter tuning. While EKF may struggle with non-linear systems, PF is more robust due to its ability to represent uncertainty using particles.

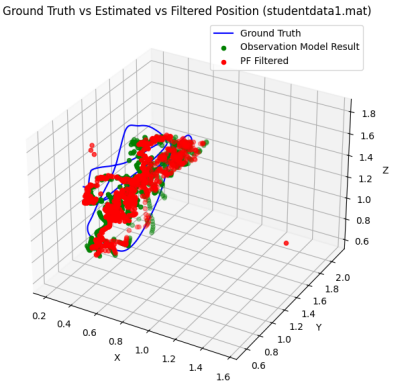
D. Conclusion

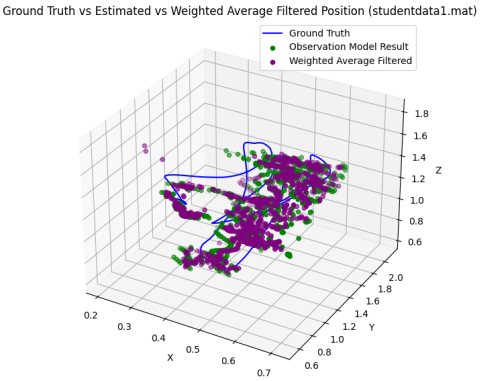
For critical applications demanding high accuracy, PF may be preferred despite higher computational cost. However, for simpler systems where efficiency is crucial, EKF may suffice.

Dataset 1

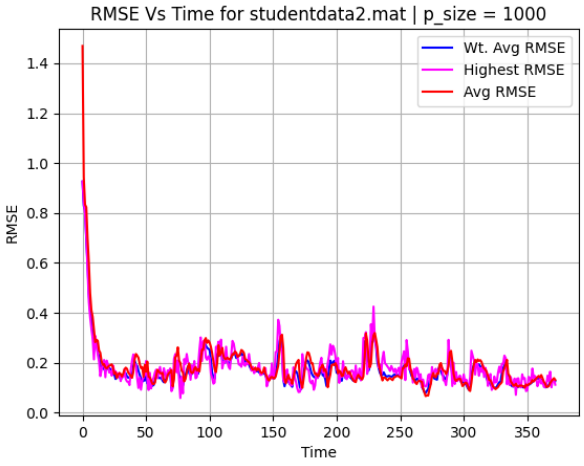

RMSE Vs Time for studentdata1.mat | p_size = 1000
Wt. Avg RMSE
Highest RMSE
Avg RMSE
RMSE
Time

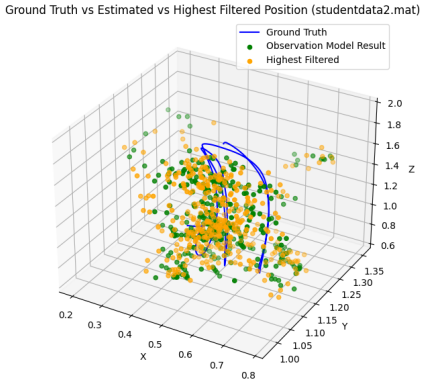

Ground Truth vs Estimated vs Highest Filtered Position (studentdata1.mat)
Ground Truth
Observation Model Result
Highest Filtered
X
Y
Z

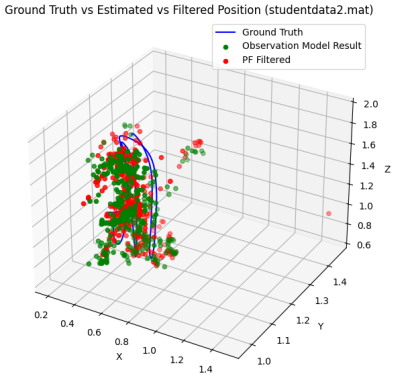

Ground Truth vs Estimated vs Filtered Position (studentdata1.mat)
Ground Truth
Observation Model Result
PF Filtered
X
Y
Z

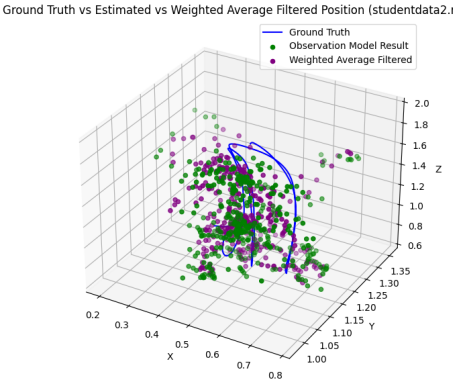

Ground Truth vs Estimated vs Weighted Average Filtered Position (studentdata1.mat)
Ground Truth
Observation Model Result
Weighted Average Filtered
X
Y
Z

Dataset 2

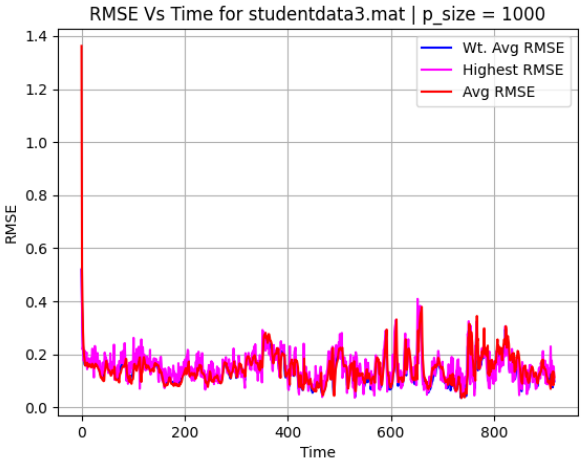

RMSE Vs Time for studentdata2.mat | p_size = 1000
Wt. Avg RMSE
Highest RMSE
Avg RMSE
RMSE
Time

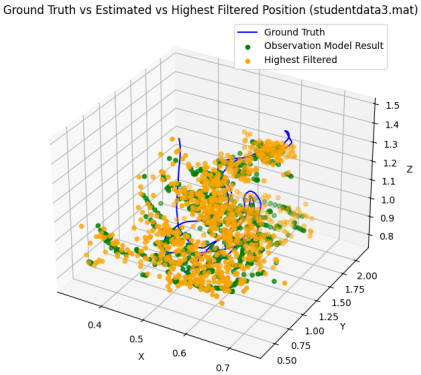

Ground Truth vs Estimated vs Highest Filtered Position (studentdata2.mat)
Ground Truth
Observation Model Result
Highest Filtered
X
Y
Z

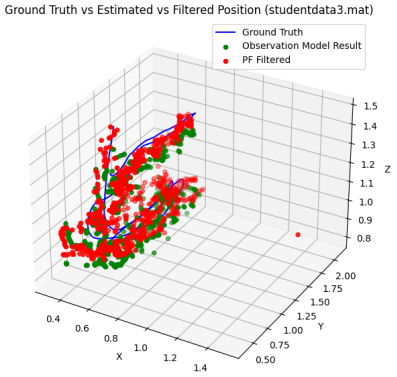

Ground Truth vs Estimated vs Filtered Position (studentdata2.mat)
Ground Truth
Observation Model Result
PF Filtered
X
Y
Z

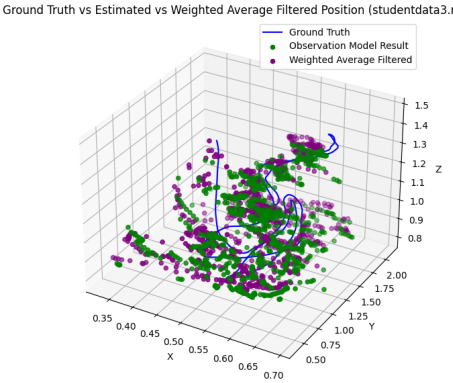

Ground Truth vs Estimated vs Weighted Average Filtered Position (studentdata2.mat)
Ground Truth
Observation Model Result
Weighted Average Filtered
X
Y
Z

Dataset 3

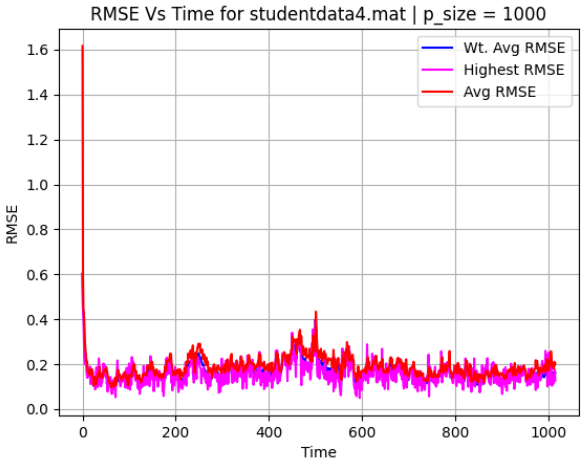

RMSE Vs Time for studentdata3.mat | p_size = 1000
Wt. Avg RMSE
Highest RMSE
Avg RMSE
RMSE
Time

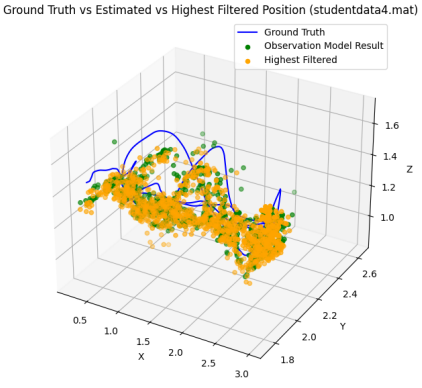

Ground Truth vs Estimated vs Highest Filtered Position (studentdata3.mat)
Ground Truth
Observation Model Result
Highest Filtered
X
Y
Z

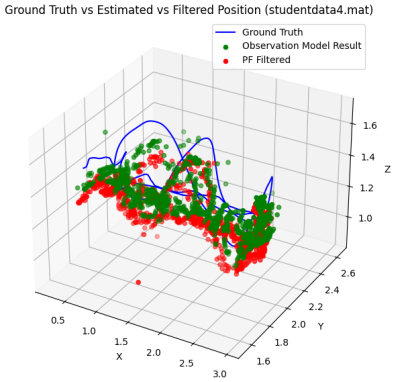

Ground Truth vs Estimated vs Filtered Position (studentdata3.mat)
Ground Truth
Observation Model Result
PF Filtered
X
Y
Z

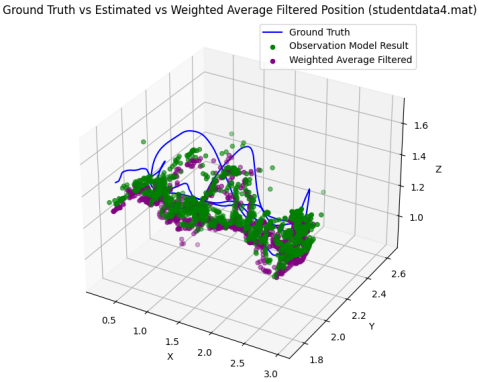

Ground Truth vs Estimated vs Weighted Average Filtered Position (studentdata3.mat)
Ground Truth
Observation Model Result
Weighted Average Filtered
X
Y
Z

Dataset 4

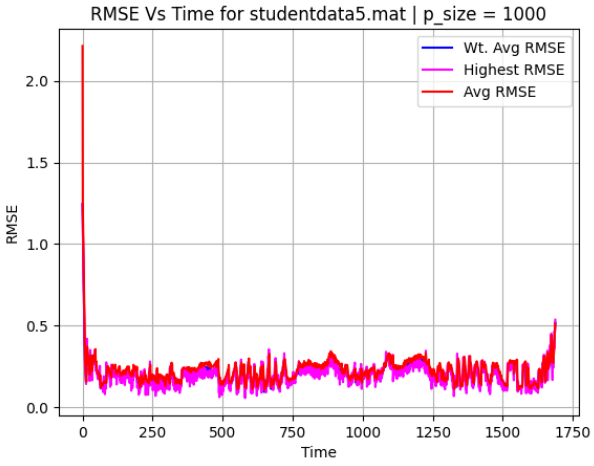

RMSE Vs Time for studentdata4.mat | p_size = 1000
Wt. Avg RMSE
Highest RMSE
Avg RMSE
RMSE
Time

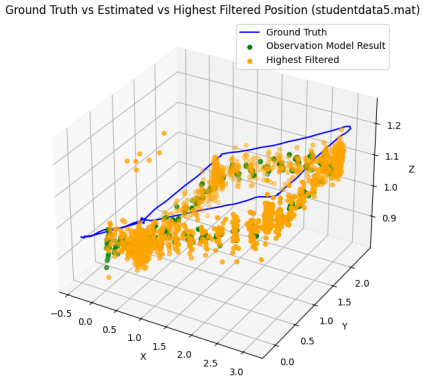

Ground Truth vs Estimated vs Highest Filtered Position (studentdata4.mat)
Ground Truth
Observation Model Result
Highest Filtered
X
Y
Z

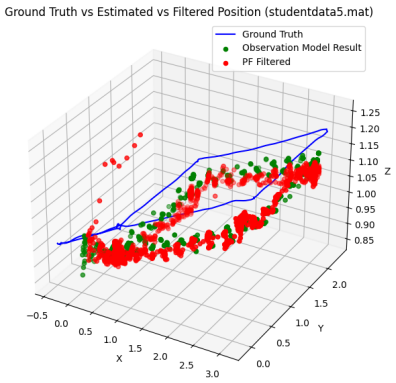

Ground Truth vs Estimated vs Filtered Position (studentdata4.mat)
Ground Truth
Observation Model Result
PF Filtered
X
Y
Z

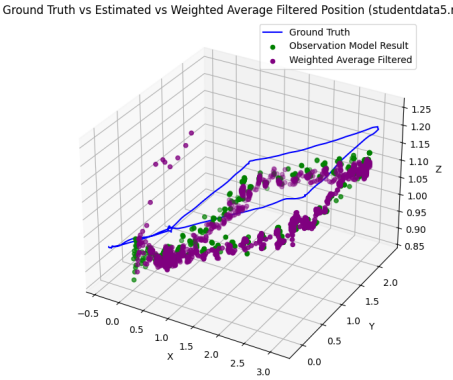

Ground Truth vs Estimated vs Weighted Average Filtered Position (studentdata4.mat)
Ground Truth
Observation Model Result
Weighted Average Filtered
X
Y
Z

Dataset 5

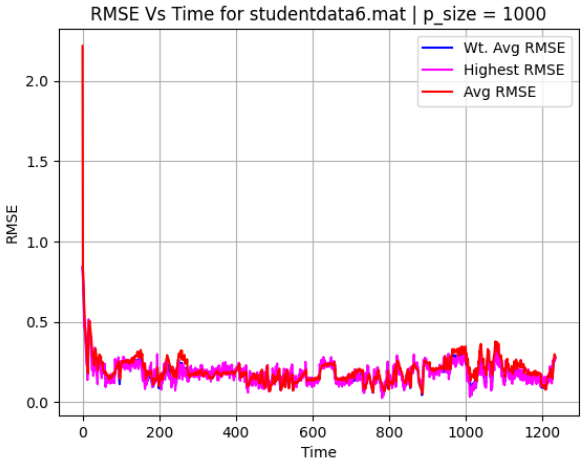

RMSE Vs Time for studentdata5.mat | p_size = 1000
Wt. Avg RMSE
Highest RMSE
Avg RMSE
RMSE
Time

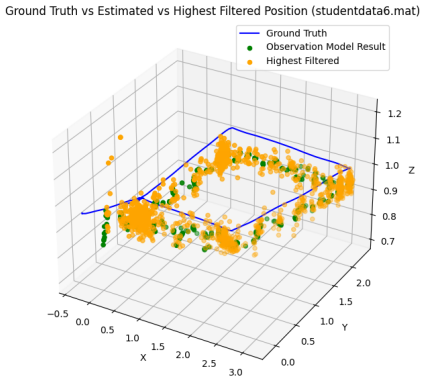

Ground Truth vs Estimated vs Highest Filtered Position (studentdata5.mat)
Ground Truth
Observation Model Result
Highest Filtered
X
Y
Z

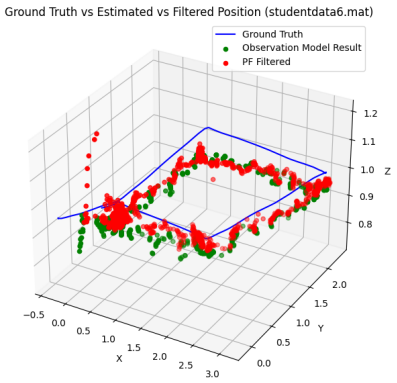

Ground Truth vs Estimated vs Filtered Position (studentdata5.mat)
Ground Truth
Observation Model Result
PF Filtered
X
Y
Z

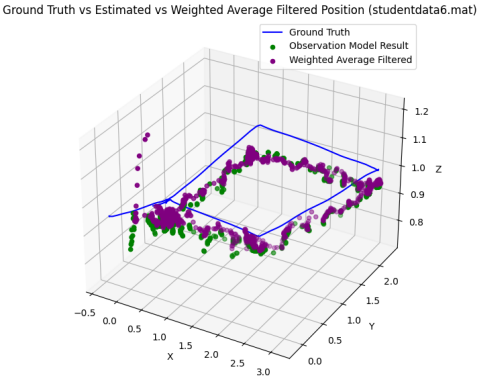

Ground Truth vs Estimated vs Weighted Average Filtered Position (studentdata5.mat)
Ground Truth
Observation Model Result
Weighted Average Filtered
X
Y
Z

Dataset 6

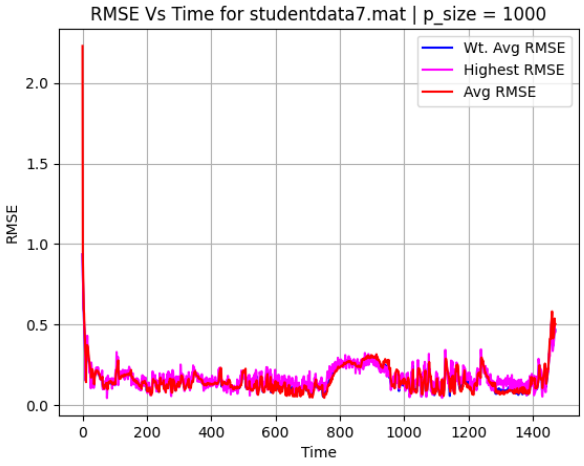

RMSE Vs Time for studentdata6.mat | p_size = 1000
Wt. Avg RMSE
Highest RMSE
Avg RMSE
RMSE
Time

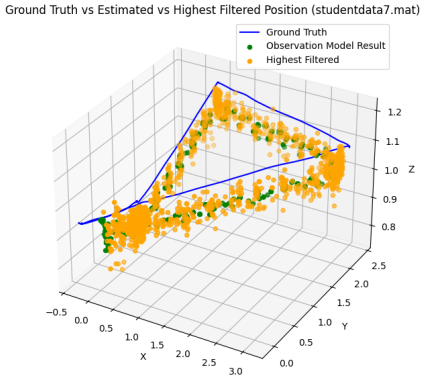

Ground Truth vs Estimated vs Highest Filtered Position (studentdata6.mat)
Ground Truth
Observation Model Result
Highest Filtered
X
Y
Z

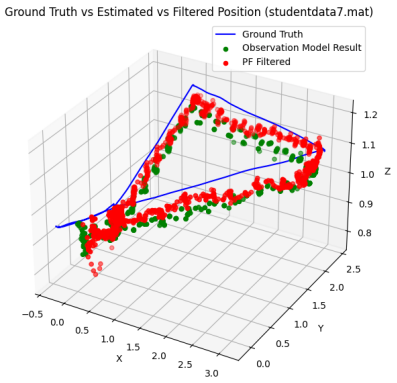

Ground Truth vs Estimated vs Filtered Position (studentdata6.mat)
Ground Truth
Observation Model Result
PF Filtered
X
Y
Z


Ground Truth vs Estimated vs Weighted Average Filtered Position (studentdata6.mat)
Ground Truth
Observation Model Result
Weighted Average Filtered
X
Y
Z

Dataset 7


RMSE Vs Time for studentdata7.mat | p_size = 1000
Wt. Avg RMSE
Highest RMSE
Avg RMSE
RMSE
Time


Ground Truth vs Estimated vs Highest Filtered Position (studentdata7.mat)
Ground Truth
Observation Model Result
Highest Filtered
X
Y
Z


Ground Truth vs Estimated vs Filtered Position (studentdata7.mat)
Ground Truth
Observation Model Result
PF Filtered
X
Y
Z


Ground Truth vs Estimated vs Weighted Average Filtered Position (studentdata7.mat)
Ground Truth
Observation Model Result
Weighted Average Filtered
X
Y
Z